Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) A method of watermarking a software object comprising the steps of:

(a) providing an input sequence; and

(b) storing a watermark in the state of the software object so that the watermark becomes detectable when ~~as~~ the software object is being run with the input sequence.

B1

2. (amended) The method as claimed in claim 1, wherein the software object is a program or a piece of a program.

3. (currently amended) The method as claimed in claim 1 wherein the watermark is detectable in the state of the software object ~~corresponds to~~ formed by the current values held in at least one of:

(a) at least one stack;

(b) at least one ~~a~~ heap;

(c) at least one data register; and

(d) at least one global variables;

of the software object.

4. (amended) The method of claim 1 or 2 or 3 wherein the watermark is stored in an execution state of the software object whereby the input sequence is constructed which, when fed to an application of which the software object is a part, will make the software object enter a second state which is a representation of the watermark, the representation being validated or checked by examining the stack, heap, global variables, registers, program counter and the like, of the software object.

5. (amended) The method of claim 1 [or 2] wherein the watermark is embedded in an execution trace of the software object whereby, as a special input is fed to the software object, an address/operator trace is monitored and, based on a property of the trace, the watermark is extracted.

B1

6. (amended) The method of claim 1 or 2 or 3 wherein the watermark is embedded in a topology of a dynamically built graph structure.

7. (currently amended) The method of claim 6 wherein the dynamically built graph structure ~~corresponds to a representation of~~ is detectable in a data structure of the program ~~and may be viewed as a set of nodes together with a set of vertices~~.

8. (currently amended) The method of claim 1, or 2 or 3 further comprising the step of:

(c) building a recognizer ~~concurrently with the input sequence and the watermark~~ operable to examine the stae of the software object when run with an input sequence and indicate whether the watermark is detectable in the state of the software object.

9. (amended) The method of claim 8 wherein the recognizer is a function adapted to identify and extract the watermark from all other dynamic structures on a heap or stack.

10. (amended) The method of claim 8 wherein the watermark incorporates a marker that will allow the recognizer to recognize it easily.

B1

11. (amended) The method of claim 8 wherein the recognizer is retained separately from the program and whereby the recognizer inspects the state of the program.

12. (amended) The method of claim 8 wherein the recognizer is dynamically linked with the program when it is checked for the existence of a watermark.

13. (amended) The method of claim 1, or 2, or 3 wherein the software object is a part of an application that is obfuscated or incorporates tamper-proofing code.

14. (amended) The method of claim 8 wherein the recognizer checks the watermark for a signature property.

15. (amended) The method of claim 14 wherein the signature property is evaluated by testing for a specific result from a hard computational problem.

16. (currently amended) The method of claim 14, wherein the watermark is embedded in a topology of a dynamically built graph structure, the method including the step of:

(d) creating a number having at least one numeric property which is ~~embedded in the topology of the watermark~~ an index of the embedded dynamically built graph structure, whereby the signature property is evaluated by testing the at least one numeric property.

B1

17. (amended) The method of claim 16 wherein the signature property is evaluated by testing whether the number is a product of two primes.

18. (currently amended) The method of varying the integrity or origin of a program including the steps of:

(a) watermarking the program with a watermark, wherein the watermark is stored in the state of a program so that the watermark becomes detectable when ~~as~~ the program is being run with an input sequence;

(b) building a recognizer, ~~concurrently with the input sequence and watermark W~~ wherein the recognizer is adapted to extract the watermark from ~~other~~ dynamically allocated data wherein the recognizer is kept separately from the program; wherein the recognizer is adapted to check for a number.

19. (currently amended) The method of claim 18, wherein the number is the product of two primes and wherein the number is ~~embedded in the topology of the watermark~~ the index of an embedded watermark graph in an enumeration of a class of possibly-embedded watermark graphs each having a different topology.

20. (currently amended) The method of claim 18 ~~or claim 19~~ wherein the number is derived from a combination of three or more prime numbers ~~depending on a context and an application for the watermark~~.

B1

21. (currently amended) The method of claim 18 or 19 wherein the program is further adapted to be resistant to tampering, the resistance to tampering ~~capable of being~~ by means of obfuscation before or after compilation, or by adding tamper-proof code.

22. (amended) The method of claim 18 or 19 wherein the recognizer checks for the effect of the watermark on an execution state of the program, thereby preserving an ability to recognize the watermark where semantics-preserving transformations have been applied to the program.

23. (currently amended) A method of watermarking software including the steps of:

(a) embedding a watermark in ~~a static~~ string; and

(b) ~~applying an obfuscation technique whereby the static string is converted into executable code~~ including in the software code that, when executed with at least one predetermined input reproduces the string of step (a), and that produces at least one other string when executed with at least one other input.

24. (currently amended) A method of watermarking software comprising the steps of:

(a) choosing a watermark ~~W~~ from a class of graphs having a plurality of members, each member of the class of graphs ~~has~~ having at least one property, the at least one property being capable of being tested by integrity-testing software; and

(b) storing the chosen ~~applying the~~ watermark ~~to~~ in the software so that when the software is run with a predetermined input sequence, the chosen watermark is reproduced.

25. (currently amended) The method of claim 24 wherein the watermark is rendered tamperproof to certain transformations by subjecting the watermark graph to ~~one or more transformations~~ redundant edge insertion.

26. (currently amended) The method of claim ~~25~~ 24 wherein the watermark is a watermark graph including at least one node and wherein each of the at least one node of the watermark graph is expanded into a cycle.

27. (currently amended) A method of fingerprinting software comprising the steps of:

(a) providing a plurality of watermarked programs, ~~the plurality of watermarked programs being obtained by providing an input sequence for each program of the plurality of~~

~~programs and storing~~ having a watermark stored in a state of a software object for the program, the watermark being detectable as the software object is being run with ~~the~~ a particular input sequence.

28. (amended) The method of fingerprinting software as claimed in claim 27 wherein the plurality of watermarked programs each of which has a number with a common prime factor.

29. (Cancelled)

30. (currently amended) A computer readable medium including a program for watermarking a software object, the program including instructions for:

B1

(a) ~~providing~~ receiving an input sequence; and

(b) storing a watermark in the state of the software object so that the watermark becomes detectable when ~~as~~ the software object is being run with the input sequence.

31. (currently amended) A computer comprising:

a software object;

means to receive an input sequence; and

a watermark stored in the state of the software object so that the watermark becomes detectable when ~~as~~ the software object is being run with ~~the~~ a particular input sequence.

32. (currently amended) A method of fingerprinting software comprising the steps of:

(a) providing a plurality of watermarked programs, the plurality of watermarked programs being obtained by watermarking each program of the plurality of programs with a watermark, wherein the watermark is stored in the state of a program so that the watermark becomes detectable when ~~as~~ the program is being run with an input sequence and building a recognizer ~~concurrently with the input sequence and watermark W~~ wherein the recognizer is adapted to extract the watermark from other dynamically allocated data wherein the recognizer is kept separately from the program; wherein the recognizer is adapted to check for a number.

B1

33. (currently amended) A method of fingerprinting software comprising the steps of:

(a) providing a plurality of watermarked programs, the plurality of watermarked programs being obtained by watermarking each program of the plurality of programs with a watermark, the watermark being obtained by embedding a watermark in ~~a static~~ string and ~~applying an obfuscation technique whereby the static string is converted into executable code~~ including in the software code that, when executed at least one predetermined input reproduces the string, and that produces at least one other string when executed with at least one other input.

34. (original) A method of fingerprinting software comprising the steps of:

(a) providing a plurality of watermarked programs, the plurality of watermarked programs being obtained by watermarking each program of the plurality of programs with a watermark, the watermark being obtained by choosing a watermark from a class of graphs having a plurality of members, each member of the class of graphs has at least one property, the at least one property being capable of being tested by integrity-testing software and applying the watermark to the software.

35. (currently amended) A computer-readable medium including a program for verifying at least one of the integrity ~~or~~ and origin of a program, the program including instructions for:

(a) watermarking the program with a watermark, wherein the watermark is stored in the state of a program so that the watermark becomes detectable when ~~as~~ the program is being run with an input sequence;

B1

(b) building a recognizer ~~concurrently with the input sequence and watermark~~ wherein the recognizer is adapted to extract the watermark from other dynamically allocated data wherein the recognizer is kept separately from the program; wherein the recognizer is adapted to check for a number.

36. (currently amended) A computer-readable medium including a program for watermarking software, the program including instructions for:

(a) embedding a watermark in a static string; and

(b) ~~applying an obfuscation technique whereby the static string is converted into executable code~~ including in the software code that, when executed with at least one predetermined input reproduces the string of step (a), and that produces at least one other string when executed with at least one other input.

37. (original) A computer-readable medium including a program for watermarking software, the program including instructions for:

(a) choosing a watermark from a class of graphs having a plurality of members, each member of the class of graphs has at least one property, the at least one property being capable of being tested by integrity-testing software; and

(b) applying the watermark to the software.

B1

38. (currently amended) A computer capable of verifying at least one of the integrity ~~or~~ and origin of a program, the computer comprising:

an input sequence;

a watermark for watermarking the program, wherein the watermark is stored in the state of a program so that the watermark becomes detectable when ~~as~~ the program is being run with the input sequence; a recognizer, ~~built concurrently with the input sequence and watermark~~ wherein the recognizer is adapted to extract the watermark from other dynamically allocated data wherein the recognizer is kept separately from the program, wherein the recognizer is adapted to check for a number.

39. (currently amended) A computer for watermarking software comprising:

(a) a ~~static~~ string;

(b) a watermark embedded in the ~~static~~ string; and

(c) an obfuscation technique for ~~converting the static string into executable code~~ including in software code that, when executed with at least one predetermined input reproduces the string, and that produces at least one other string when executed with at least one other input.

B1

40. (original) A computer comprising:

a watermark from a class of graphs having a plurality of members, each member of the class of graphs has at least one property, the at least one property being capable of being tested by integrity-testing software; and

software to which the watermark is applied.

41. (new) The method of claim 1, wherein the watermark is detectable in any portion of the dynamic data state of the software object.

B2

42. (new) The method of claim 1, wherein the software object is an executable media object.

43. (new) The method of claim 1, wherein no part of the state of the software object in which the watermark becomes detectable is visually or audibly apparent.

44. (new) The method of claim 8, wherein the recognizer is built concurrently with the watermark and input sequence.

45. (new) The method of claim 16, wherein said graph is an enumeration of a class of possibly-embedded watermark graphs each having a different topology.

46. (new) The method of claim 18, wherein the recognizer is built concurrently with the watermark and input sequence.

47. (new) The method of claim 23, wherein the software code includes a state variable updated during execution of the code and that influences the string outputted.

48. (new) The method of claim 31, wherein no part of the state of the software object in which the watermark becomes detectable is visually or audibly apparent.

B2

49. (new) A method of watermarking software including the steps of:

(a) embedding a watermark in a string, thereby forming a watermark string;

(b) converting the string into executable code that, when executed, constructs a dynamic string by:

(i) receiving an input in the form of at least one input variable;

(ii) defining and updating a string index variable that controls the location of writing to the dynamic string;

(iii) defining a state variable and updating the state variable in a dynamically-determined fashion so that the value of said state variable in at least one read operation will vary depending on which of at least two execution paths was taken by said executable code to reach said operation; and

(iv) dependent on said at least one input variable, string index variable and state variable, writing to the dynamic string;

wherein the dynamic string reconstructs the watermark string for at least one predetermined input and not for the other inputs.

50. (new) A method of watermarking a software object comprising the steps of:

(a) providing an input sequence; and

(b) storing a watermark in the state of the software object as the software object is being run with the input sequence, wherein the watermark is stored in an execution state of the software object whereby the input sequence is constructed which, when fed to an application of which the software object is a part, will make the software object enter a second state which is a representation of the watermark, the representation being validated or checked by examining the stack, heap, global variables, registers, program counter and the like, of the software object.

51. (new) A method of watermarking a software object comprising the steps of:

(a) providing an input sequence; and

(b) storing a watermark in the state of the software object as the software object is being run with the input sequence, wherein the watermark is embedded in an execution trace of the software object whereby, as a special input is fed to the software object, an address/operator trace is monitored and, based on a property of the trace, the watermark is extracted.

52. (new) The method of varying the integrity or origin of a program including the steps of:

(a) watermarking the program with a watermark, wherein the watermark is stored in the state of a program as the program is being run with an input sequence;

B2

(b) building a recognizer, wherein the recognizer is adapted to extract the watermark from dynamically allocated data wherein the recognizer is kept separately from the program; wherein the recognizer is adapted to check for a number, wherein the recognizer checks for the effect of the watermark on an execution state of the program, thereby preserving an ability to recognize the watermark where semantics-preserving transformations have been applied to the program.